Chap.9 Tree-based methods [Book, Sect. 9.2]

9.1 Classification and regression trees (CART)

In the American Medical Association's Encyclopedia of Medicine, there are many tree-structured flow charts for patient diagnosis. E.g., the questions asked are: Is the body temperature above normal? Is the patient feeling pain? Is the pain in the chest area? The terminal nodes are the diseases, e.g. influenza, heart attack, food poisoning, etc. Flow chart looks like an inverted tree.

Decision tree methods partition the predictor $\mathbf{x}$-space into regions and fit a simple function $f(\mathbf{x})$ in each region.

The most common decision tree method is *CART (classification and regression tree)* (Brieman et al., 1984), which partitions the predictor space into rectangular regions, and for regression problems, fits $f(\mathbf{x})$ = constant in each region, so there is a step at the boundary between two regions.

CART is useful for two main reasons:
(i) It gives an intuitive display on how the predictand or response variable broadly depends on the predictors.
(ii) When there are many predictors, it provides a computationally inexpensive way to select a smaller number of relevant predictors.

These selected predictors can then be used in more accurate but computationally more expensive models like NN, etc., i.e. CART can be used to pre-screen predictors for more sophisticated models (Burrows, 1999), (although stepwise linear regression often does better than CART in pre-screening).

CART can be used for both nonlinear classification & regression.

Focus on the regression problem, with y_{d} the predictand data. Suppose there are two predictor variables x_1 and x_2.

Look for the partition point $x_1^{(1)}$ where the step function $f(\mathbf{x}) = c_1$ for $x_1 < x_1^{(1)}$, and $f(\mathbf{x}) = c_2$ for $x_1 \geq x_1^{(1)}$ gives the best fit to the predictand data.

If the fit is determined by the mean squared error (MSE), then the constants c_1 and c_2 are simply given by the mean value of y_{d} over the two separate regions partitioned by $x_1^{(1)}$.

A similar search for a partition point $x_2^{(1)}$ is performed in the x_2 direction. Whether our first partition should be at $x_1^{(1)}$ or $x_2^{(1)}$ is based on whichever partition yields the smallest MSE.

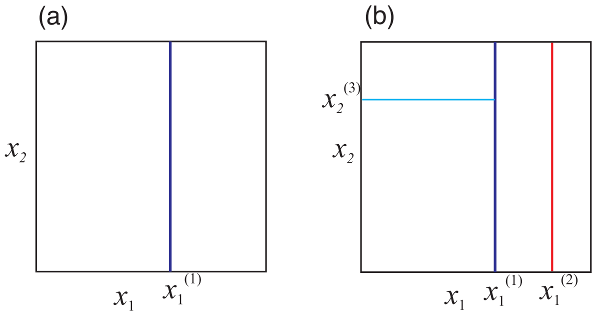


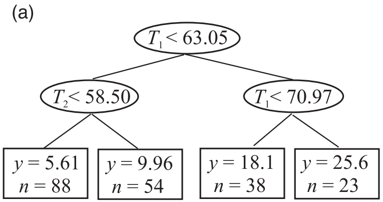
Figure : Schematic diagram illustrating the partitioning of the predictor **x**-space by CART. (a) First partition at $x_1 = x_1^{(1)}$ yields 2 regions, each with a constant value for the predictand y. (b) Second partition at $x_1 = x_1^{(2)}$ is followed by a 3rd partition at $x_2 = x_2^{(3)}$, yielding 4 regions, and the predictand described by the 4 constants over the 4 regions.

The partition process is repeated until some stopping criterion is met.

Illustrate CART with a dataset containing the daily maximum of the hourly-averaged ozone at Los Angeles, with high ozone level indicating poor air quality.

Nine predictor variables for the ozone, including temperature measurements T_1 and T_2 at two stations, visibility measured at the Los Angeles airport, and the pressure gradient between the airport and another station.

In the 9-dimensional predictor space, CART made the 1st partition at $T_1 = 63.05^\circ$F, 2nd partition at $T_1 = 70.97^\circ$F and 3rd partition at $T_2 = 58.50^\circ$F. The partitioned regions are shown schematically in Fig.b above, with x_1 & x_2 representing T_1 & T_2, respectively.



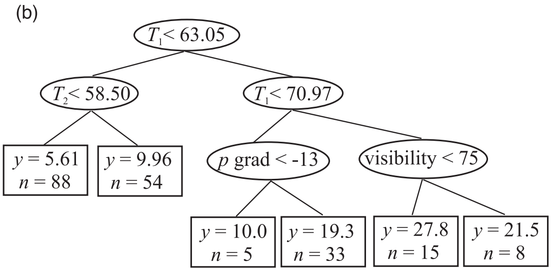


Figure : Regression tree from CART where the predictand y is the Los Angeles ozone level (in ppm), and there are nine predictor variables. The “tree” is plotted upside down, with the “leaves” (i.e. terminal nodes) drawn as rectangular boxes at the bottom and the non-terminal nodes as ellipses. (a) The tree after 3 partitions has 4 leaf nodes, while (b) the tree after 5 partitions has 6 leaf nodes. In each ellipse, a condition is given. Starting from the top ellipse, if the condition is satisfied, proceed along the left branch down to the next node; if not, proceed along the right branch; continue until a leaf node is reached. In each rectangular box, the constant value of y (computed from the mean of the data y_{d}) in the partitioned region associated with the particular leaf node is given, as well as n, the number of data points in that region. Among the nine predictor variables, the most relevant ones are the temperatures T_1 and T_2 (in $^\circ$F) at two stations, p grad (pressure gradient in mm Hg) and visibility (in miles).

If one continues partitioning, the tree grows further. In Fig. b, there is now a fourth partition at pressure gradient $= -13$ mm Hg, and a fifth partition at visibility $= 75$ miles. Hence CART tells us that the most important predictors, in decreasing order of importance, are T_1, T_2, pressure gradient and visibility.

The tree now has six terminal or leaf nodes, denoting the six regions formed by the partitions. Each region is associated with a constant ozone value, the highest being 27.8 ppm (attained by the second leaf from the right).

From this leaf node, one can then retrace the path towards the root, which tells us that this highest ozone leaf node was reached after satisfying first $63.05^\circ\text{F} \leq T_1$, then $70.97^\circ\text{F} \leq T_1$ and finally visibility < 75 miles, i.e. the highest ozone conditions tend to occur at high temperature and low visibility.

Q1: What are the conditions for the lowest ozone values to occur?

CART also give the number data points in each partitioned region, e.g. 15 points belong to the highest ozone leaf node versus 88 to the lowest ozone node, out of a total of 203 data points.

After training is done, when a new value of the predictor **x** becomes available, one proceeds along the tree till a leaf node is reached, and the forecast for y is then simply the constant value associated with that leaf node.

How big a tree should one grow? It is not a good idea to stop the growing process after encountering a partition which gave little improvement in the MSE, because a further partition may lead to a large drop in the MSE.

Instead, one lets the tree grow to a large size, and uses regularization (i.e. weight penalty) to prune the tree down to the optimal size.

Let $L =$ the number of leaf nodes. A regularized objective function is

$$J(L) = E(L) + PL, \tag{1}$$

where $E(L)$ is the MSE for the tree with L leaf nodes, and P is the weight penalty parameter, penalizing trees with excessive leaf nodes.

The process to generate a sequence of trees with varying L is:
Start with the full tree, remove the internal node the demise of which leads to the smallest increase in MSE, and continue until the tree has only one internal node.

From this sequence of trees with a wide range of L values, one chooses the tree with the smallest $J(L)$, thus selecting the tree with the optimal size for a given P.
The best value for P is determined from cross-validation.

CART can also be used for classification. The constant value for y over a region is now replaced by the class k to which the largest number of y_d belong. Instead of MSE, we need a different E.

Let p_{lk} denoting the proportion of data in region l belonging to class k. Ideally, p_{lk} should be as close to 0 as possible except for one class, i.e. each region is ideally occupied by only one class, so aim for $p_{lk} = 0$ or 1. Two common choices for E:
The Gini index

$$E(L) = \sum_{l=1}^{L} \sum_{k=1}^{K} p_{lk}(1 - p_{lk})\,, \tag{2}$$

or cross entropy

$$E(L) = -\sum_{l=1}^{L}\sum_{k=1}^{K} p_{lk} \ln p_{lk} \,, \tag{3}$$

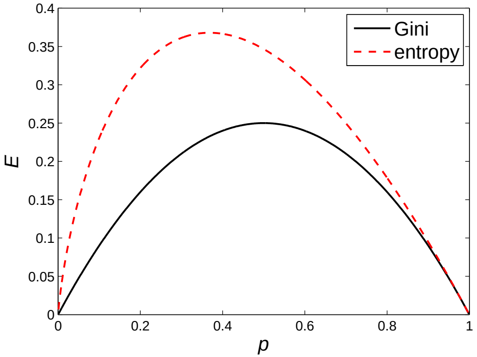


Figure : Error E being $p(1-p)$ (Gini index) and $-p\ln p$ (cross entropy).

So far, the partitions are of the form $x_i < s$, thus restricting the partition boundaries to lie parallel to the axes in **x**-space. If a decision boundary in the x_1-x_2 plane is oriented at 45° to the x_1 axis, then it would take many parallel-axes partitions to approximate such a decision boundary.

Some versions of CART allow partitions of the form $\sum_i a_i x_i < s$, which are not restricted to be parallel to the axes, but the easy interpretability of CART is lost.

CART is not very stable: A small change in the data can result in a very different series of splits.
This instability is due to the hierarchical nature of the process, i.e. the effect of an error in the top split is propagated down to all of the splits below it.

CART has low predictions skills relative to NN.

CART uses piecewise constant functions to represent the nonlinear relation $y = f(\mathbf{x})$. To extend beyond a zero-order model like CART, first-order models, like the M5 tree model (Quinlan, 1993), use piecewise linear functions to represent the nonlinear relation.

9.2 Random forests (RF)

The unstable nature and low prediction skills of CART are overcome by using an ensemble of trees. A boostrap ensemble of CART models is called a random forest (Breiman, 2001):

If N is the number of training data points and M the number of predictor variables, one generates many bootstrap samples (by selecting N data points with replacement from the training dataset), then train CART on each bootstrap sample using m randomly chosen predictors out of the original M predictors ($m \ll M$ if M is large). The trees are fully grown without pruning.

With new predictor data, y is taken to be the mean of the ensemble output in regression problems, or the class k chosen by the largest number of ensemble members in classification problems.

Choosing randomly $m \ll M$ predictors ensures ensemble members are very different from each other $=>$ outperforms RF with $m = M$.

When the number of predictors is large, but the fraction of relevant predictors is small, RF is likely to perform poorly with small m since many trees will not have relevant predictors.

Default m is about $M/3$ for regression and $\sqrt{M}$ for classification, and minimum number of observations in a leaf node is 5 for regression and 1 for classification.
Best to determine these hyperparameters, esp. m, by validation.

Data not selected during the bootstrap resampling, i.e. the out-of-bag samples, can be used for validation.

An alternative to RF is an ensemble of boosted trees, i.e. boosting with trees. Hastie et al. (2009, Chap.15) shows boosted trees outperforming RF. Caruana and Niculescu-Mizil (2006) shows boosted trees outperforming RF, NN etc.

Matlab codes:

CART is classregtree.m from the MATLAB statistical toolbox.

`www.mathworks.com/help/toolbox/stats/classregtree.html`
t = classregtree(X, y, ...)

Random forest code is downloadable from
`http://code.google.com/p/randomforest-matlab/`

MATLAB Statistical Toolbox has its own random forest code, named treebagger:
`www.mathworks.com/help/toolbox/stats/treebagger.html`
but from what I've read, it may be slower than randomforest-matlab.

M5 tree is available from the Weka package,
`weka.sourceforge.net/doc/weka/classifiers/trees/m5/M5Base.html`

Boosting (with trees):
`http://www.mathworks.com/help/stats/fitensemble.html`

References:

Breiman, L. (2001). Random forests. *Machine Learning*, 45:5–32.

Brieman, L., Friedman, J., Olshen, R. A., and Stone, C. (1984). *Classification and Regression Trees*. Chapman and Hall, New York.

Burrows, W. R. (1999). Combining classification and regression trees and the neuro-fuzzy inference system for environmental data modeling. In *18th International Conference of the North American Fuzzy Information Processing Society - NAFIPS*, pages 695–699, New York, NY.

Caruana, R. and Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 161–68, Pittsburgh, PA.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, 2nd edition.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo.